Distributed Database(DDB)

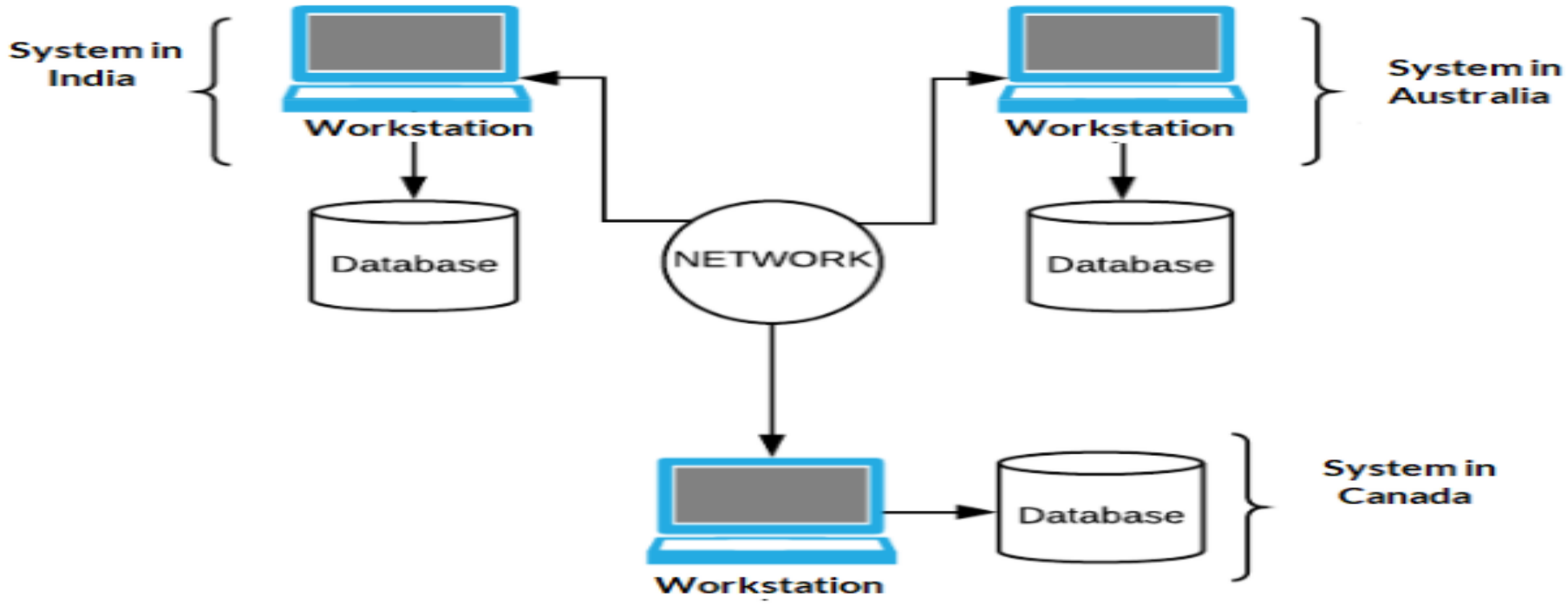
Zain Arshad
UOL Defence Road
zain.arshad@cs.uol.edu.pk

Lecture 11

Architectural models of Distributed Databases

Distributed database management system

- In a distributed database management system, the database is not stored at a single location. Rather, it may be stored in multiple computers at the same place or geographically spread far away. Despite all this, the distributed database appears as a single database to the user.



Distributed database management system

- As seen in the figure, the components of the distributed database can be in multiple locations such as India, Canada, Australia, etc.
- However, this is transparent to the user i.e the database appears as a single entity

Advantages of Distributed Databases

- Following are the advantages of distributed databases over centralized databases.

- **Modular Development** − If the system needs to be expanded to new locations or new units, in centralized database systems, the action requires substantial efforts and disruption in the existing functioning. However, in distributed databases, the work simply requires adding new computers and local data to the new site and finally connecting them to the distributed system, with no interruption in current functions.

- **More Reliable** − In case of database failures, the total system of centralized databases comes to a halt. However, in distributed systems, when a component fails, the functioning of the system continues may be at a reduced performance. Hence DDBMS is more reliable.

Advantages of Distributed Databases

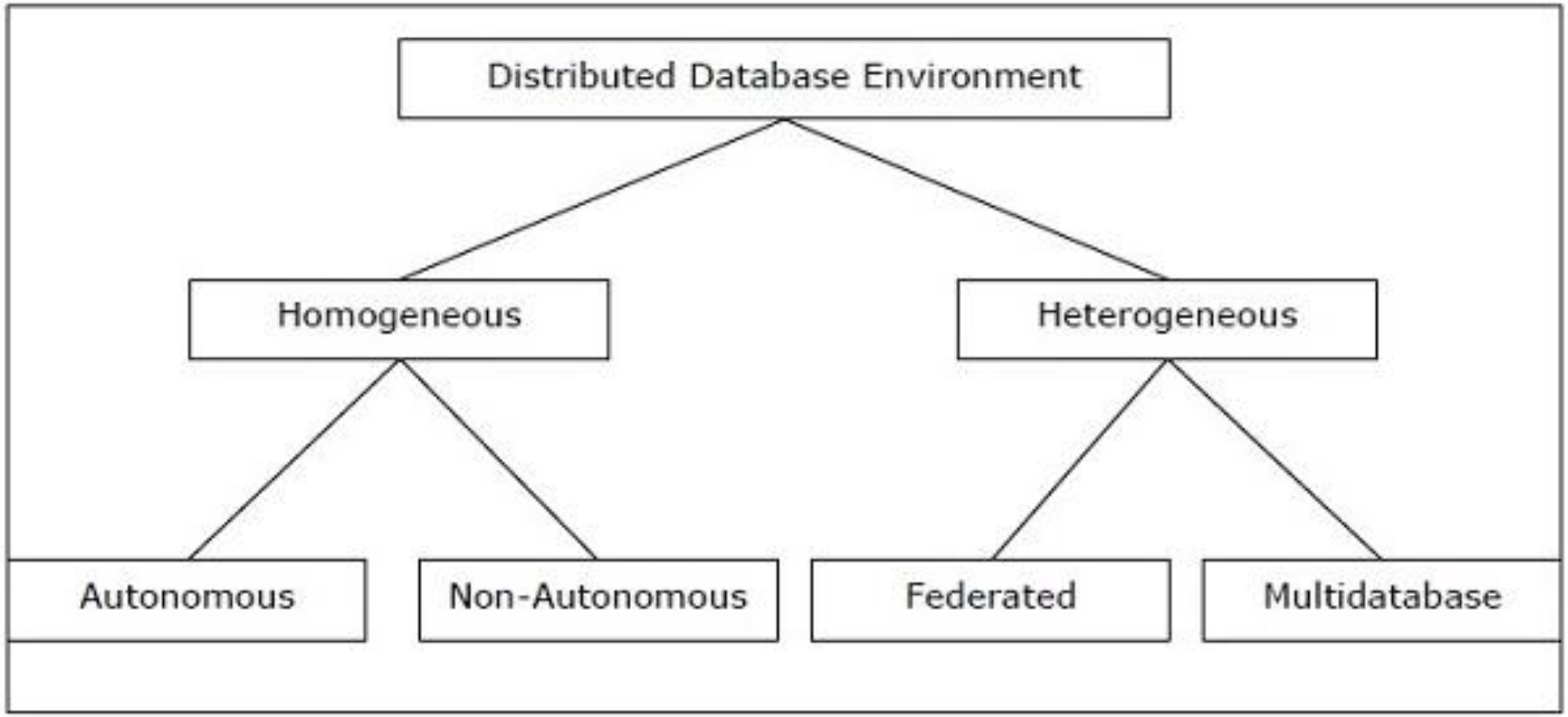
- **Better Response** − If data is distributed in an efficient manner, then user requests can be met from local data itself, thus providing faster response. On the other hand, in centralized systems, all queries have to pass through the central computer for processing, which increases the response time.

- **Lower Communication Cost** − In distributed database systems, if data is located locally where it is mostly used, then the communication costs for data manipulation can be minimized. This is not feasible in centralized systems.

Challenges of Distributed Databases

- Following are some of the adversities associated with distributed databases.
- **Need for complex and expensive software** − DDBMS demands complex and often expensive software to provide data transparency and co-ordination across the several sites.
- **Processing overhead** − Even simple operations may require a large number of communications and additional calculations to provide uniformity in data across the sites.
- **Data integrity** − The need for updating data in multiple sites pose problems of data integrity.
- **Overheads for improper data distribution** − Responsiveness of queries is largely dependent upon proper data distribution. Improper data distribution often leads to very slow response to user requests.

Types of Distributed Databases



Types Of Distributed Databases

•Homogeneous DDB:

In a homogeneous database, all different sites store database identically.

• The operating system, database management system and the data structures used – all are same at all sites. Hence, they're easy to manage.

Types:

1. **Autonomous**
2. **Non autonomous**

Types of Homogeneous Databases

1. **Autonomous** − Each database is independent that functions on its own. They are integrated by a controlling application and use message passing to share data updates.

2. **Non-autonomous** − Data is distributed across the homogeneous nodes and a central or master DBMS co-ordinates data updates across the sites.

Heterogeneous DDB

- In a heterogeneous distributed database, different sites can use different schema and software that can lead to problems in query processing and transactions. Also, a particular site might be completely unaware of the other sites.
- Different computers may use a different operating system, different database application. They may even use different data models for the database. Hence, translations are required for different sites to communicate.

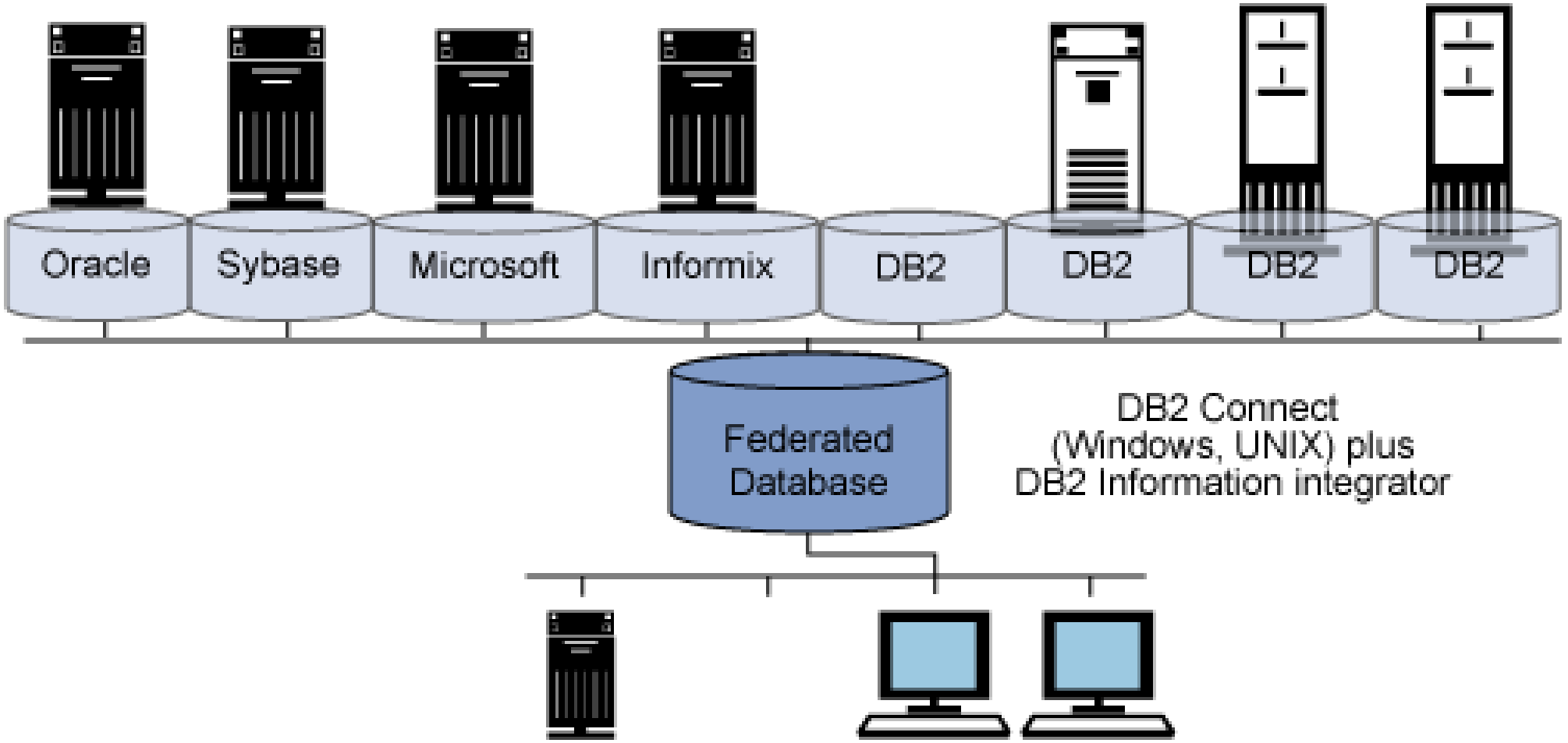
Types:

1. **Federated**
2. **Un-Federated**

Types of Heterogeneous Databases

- **Federated** − A federated database is a system in which several databases appear to function as a single entity. Each component database in the system is completely self-sustained and functional. When an application queries the federated database, the system figures out which of its component databases contains the data being requested and passes the request to it. Federated databases can be thought of as database virtualization in much the same way that storage virtualization makes several drives appear as one.
- **Un-federated** − The database systems employ a central coordinating module through which the databases are accessed.

Federated Database



DDBMS Parameters

DDBMS architectures are generally developed depending on three parameters −

1. **Distribution**
2. **Autonomy**
3. **Heterogeneity**

1. **Distribution** − It states the physical distribution of data across the different sites.
2. **Autonomy** − It indicates the distribution of control of the database system and the degree to which each constituent DBMS can operate independently.
3. **Heterogeneity** − It refers to the uniformity or dissimilarity of the data models, system components and databases.

Architectural Models

- Some of the common architectural models are −

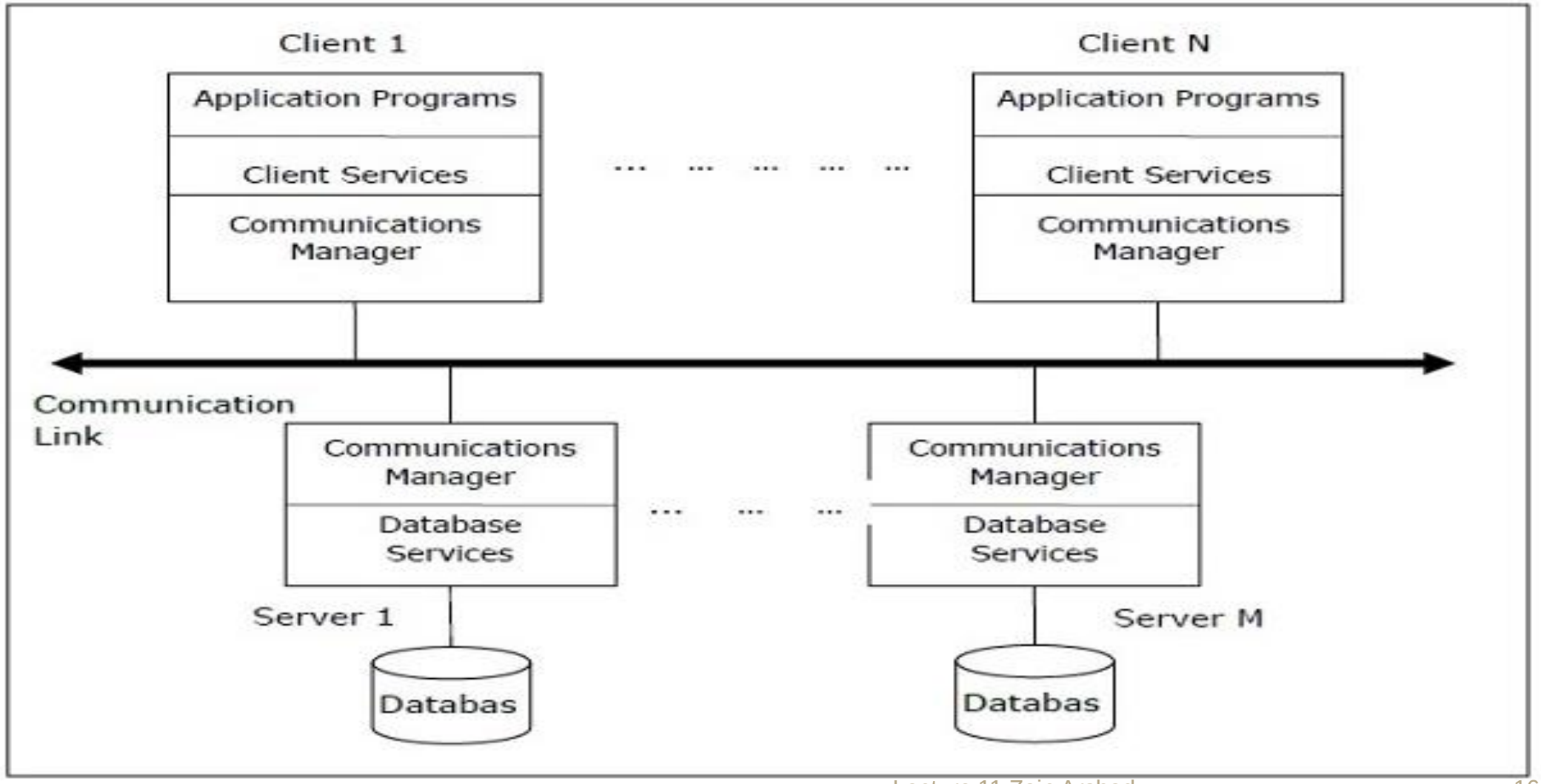
1. **Client - Server Architecture for DDBMS**
2. **Peer - to - Peer Architecture for DDBMS**
3. **Multi - DBMS Architecture**

Client - Server Architecture

- This is a two-level architecture where the functionality is divided into servers and clients. The server functions primarily encompass data management, query processing, optimization and transaction management. Client functions include mainly user interface.
- The two different client - server architecture are −

1. **Single Server Multiple Client**
2. **Multiple Server Multiple Client**

Client - Server Architecture

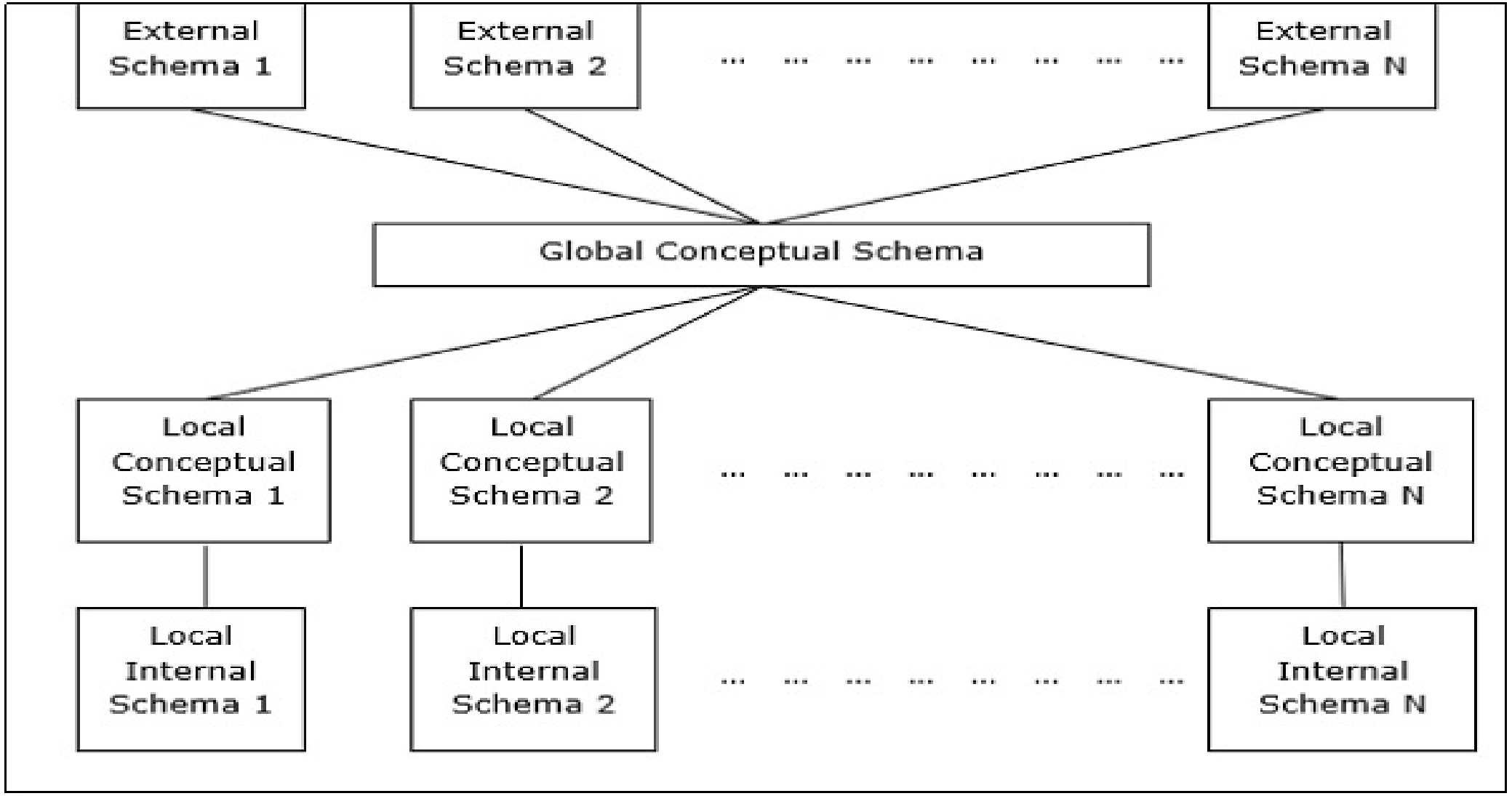


Peer- to-Peer Architecture for DDBMS

- In these systems, each peer acts both as a client and a server for imparting database services. The peers share their resource with other peers and co-ordinate their activities.
- This architecture generally has four levels of schemas −

1. **Local Internal Schema** − Depicts physical data organization at each site.
2. **Local Conceptual Schema** − Depicts logical data organization at each site.
3. **Global Conceptual Schema** − Depicts the global logical view of data.
4. **External Schema** − Depicts user view of data.

Peer- to-Peer Architecture for DDBMS



Multi - DBMS Architectures

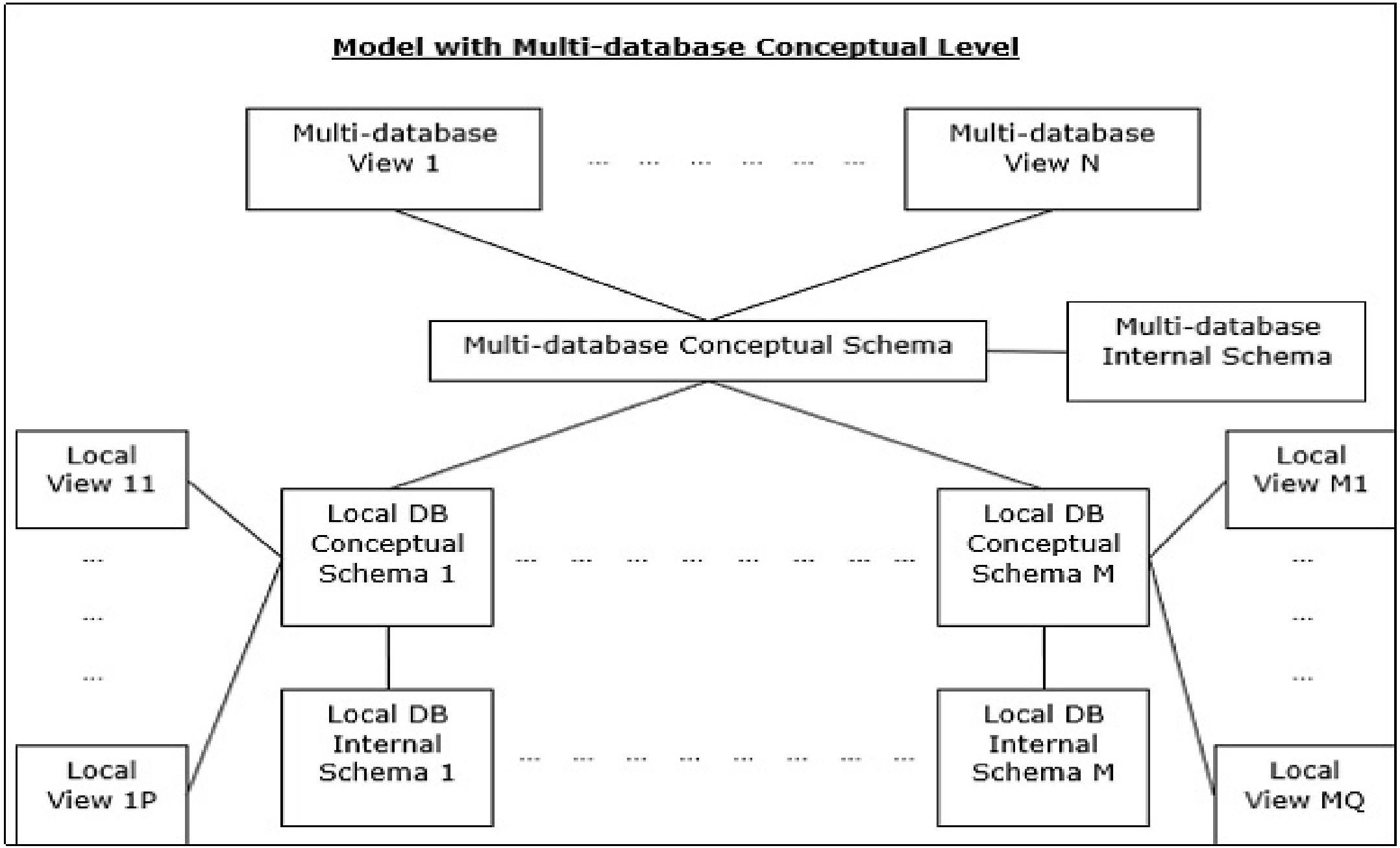
- This is an integrated database system formed by a collection of two or more autonomous database systems.
- Multi-DBMS can be expressed through six levels of schemas −

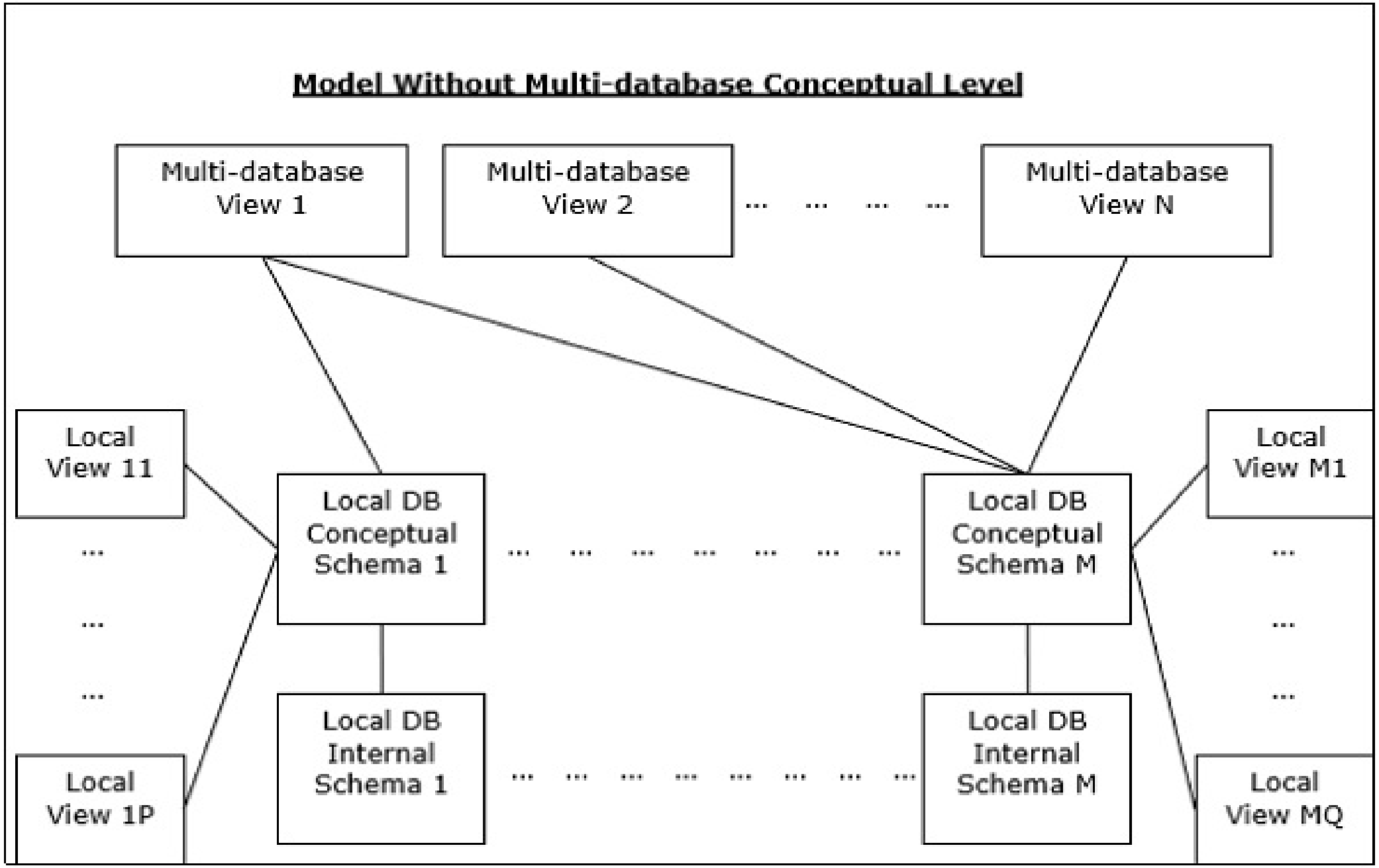
- **Multi-database View Level** − Depicts multiple user views comprising of subsets of the integrated distributed database.
- **Multi-database Conceptual Level** − Depicts integrated multi-database that comprises of global logical multi-database structure definitions.
- **Multi-database Internal Level** − Depicts the data distribution across different sites and multi-database to local data mapping.
- **Local database View Level** − Depicts public view of local data.
- **Local database Conceptual Level** − Depicts local data organization at each site.
- **Local database Internal Level** − Depicts physical data organization at each site.

Design alternatives

There are two design alternatives for multi-DBMS −

1. **Model with multi-database conceptual level.**
2. **Model without multi-database conceptual level.**


Model with Multi-database Conceptual Level
Multi-database View 1
Multi-database View N
Multi-database Conceptual Schema
Multi-database Internal Schema
Local View 11
Local DB Conceptual Schema 1
Local DB Internal Schema 1
Local View 1P
Local DB Conceptual Schema M
Local DB Internal Schema M
Local View M1
Local View MQ


Model Without Multi-database Conceptual Level
Multi-database View 1
Multi-database View 2
Multi-database View N
Local View 11
Local View 1P
Local DB Conceptual Schema 1
Local DB Internal Schema 1
Local DB Conceptual Schema M
Local DB Internal Schema M
Local View M1
Local View MQ

Design Alternatives

The distribution design alternatives for the tables in a DDBMS are as follows −

1. **Non-replicated and non-fragmented**
2. **Fully replicated**
3. **Partially replicated**
4. **Fragmented**
5. **Hybrid**

Non-replicated & Non-fragmented

- In this design alternative, different tables are placed at different sites. Data is placed so that it is at a close proximity to the site where it is used most.
- It is most suitable for database systems where the percentage of queries needed to join information in tables placed at different sites is low. If an appropriate distribution strategy is adopted, then this design alternative helps to reduce the communication cost during data processing.

Fully Replicated

- In this design alternative, at each site, one copy of all the database tables is stored. Since, each site has its own copy of the entire database, queries are very fast requiring negligible communication cost.
- On the contrary, the massive redundancy in data requires huge cost during update operations. Hence, this is suitable for systems where a large number of queries is required to be handled whereas the number of database updates is low.

Partially Replicated

- Copies of tables or portions of tables are stored at different sites. The distribution of the tables is done in accordance to the frequency of access. This takes into consideration the fact that the frequency of accessing the tables vary considerably from site to site.
- The number of copies of the tables (or portions) depends on how frequently the access queries execute and the site which generate the access queries.

Advantages of Data Replication

1. **Reliability** − In case of failure of any site, the database system continues to work since a copy is available at another site(s).
2. **Reduction in Network Load** − Since local copies of data are available, query processing can be done with reduced network usage, particularly during prime hours. Data updating can be done at non-prime hours.
3. **Quicker Response** − Availability of local copies of data ensures quick query processing and consequently quick response time.
4. **Simpler Transactions** − Transactions require less number of joins of tables located at different sites and minimal coordination across the network. Thus, they become simpler in nature.

Disadvantages of Data Replication

1. **Increased Storage Requirements** – Maintaining multiple copies of data is associated with increased storage costs. The storage space required is in multiples of the storage required for a centralized system.
2. **Increased Cost and Complexity of Data Updating** – Each time a data item is updated, the update needs to be reflected in all the copies of the data at the different sites. This requires complex synchronization techniques and protocols.
3. **Undesirable Application – Database coupling** – If complex update mechanisms are not used, removing data inconsistency requires complex co-ordination at application level. This results in undesirable application – database coupling.

Replication techniques

- Some commonly used replication techniques are −

1. **Snapshot replication**
2. **Near-real-time replication**
3. **Pull replication**

Fragmented

- In this design, a table is divided into two or more pieces referred to as fragments or partitions, and each fragment can be stored at different sites. This considers the fact that it seldom happens that all data stored in a table is required at a given site.

- Moreover, fragmentation increases parallelism and provides better disaster recovery. Here, there is only one copy of each fragment in the system, i.e. no redundant data.

Advantages of Fragmentation

- Since data is stored close to the site of usage, efficiency of the database system is increased.
- Local query optimization techniques are sufficient for most queries since data is locally available.
- Since irrelevant data is not available at the sites, security and privacy of the database system can be maintained.

Disadvantages of Fragmentation

- When data from different fragments are required, the access speeds may be very high.
- In case of recursive fragmentations, the job of reconstruction will need expensive techniques.
- Lack of back-up copies of data in different sites may render the database ineffective in case of failure of a site.

Relational Database Concepts

- A database is a **structured collection** of data related to some real-life phenomena that we are trying to model.
- As an **example** we use a database that models an **engineering company**.

- For each **employee**, we would like to keep track of the **employee number (ENO), name (ENAME), title in the company (TITLE), salary (SAL), identification number of the project(s) the employee is working on (PNO), responsibility within the project (RESP),** and **duration of the assignment to the project (DUR)** in months.

- Similarly, for each **project** we would like to store the **project number (PNO)**, the **project name (PNAME)**, and the **project budget (BUDGET)**.

Sample Database Instance

EMP

ENO	ENAME	TITLE	SAL	PNO	RESP	DUR
E1	J. Doe	Elect. Eng.	40000	P1	Manager	12
E2	M. Smith	Analyst	34000	P1	Analyst	24
E2	M. Smith	Analyst	34000	P2	Analyst	6
E3	A. Lee	Mech. Eng.	27000	P3	Consultant	10
E3	A. Lee	Mech. Eng.	27000	P4	Engineer	48
E4	J. Miller	Programmer	24000	P2	Programmer	18
E5	B. Casey	Syst. Anal.	34000	P2	Manager	24
E6	L. Chu	Elect. Eng.	40000	P4	Manager	48
E7	R. Davis	Mech. Eng.	27000	P3	Engineer	36
E8	J. Jones	Syst. Anal.	34000	P3	Manager	40

PROJ

PNO	PNAME	BUDGET
P1	Instrumentation	150000
P2	Database Develop.	135000
P3	CAD/CAM	250000
P4	Maintenance	310000

Relational Database Terms

EMP

ENO	ENAME	TITLE	SAL	PNO	RESP	DUR

PROJ

PNO	PNAME	BUDGET

Sample Database Scheme

- The entities to be modelled are the **employees** (EMP) and **projects (PROJ)**.
- The relation schemas for this database can be defined as follows:
- **EMP(ENO, ENAME, TITLE, SAL, PNO, RESP, DUR).**
- **PROJ(PNO,PNAME, BUDGET).**
- In relation scheme **EMP**, there are **seven attributes**: **ENO, ENAME, TITLE, SAL,PNO, RESP, DUR.**
- In relation scheme **PROJ**, there are **three attributes**: **PNO, PNAME, BUDGET**

Relational Database Terms

- The number of **attributes** of a relation defines its **degree**, whereas the number of **tuples** of the relation defines its **cardinality**.
- The **empty table**, showing the structure of the table, corresponds to the **relation schema**.

- When the table is **filled with rows**, it corresponds to a **relation instance**.
- Special value of the attribute is generally referred to as the **null value.**
- **Special care** should be given to **differentiate null** from **zero**.

- For example, **value "0"** for **attribute DUR** is **known information** (e.g., in the case of a newly hired employee), while value "**null**" for DUR means **unknown**.

Anomalies

- The **aim of normalization** is to **eliminate various anomalies** (or undesirable aspects) **of a relation** in order **to obtain "better" relations**.Following **four problems** might exist in a **relation scheme**.

1. **Repetition anomaly**
2. **Update anomaly.**
3. **Insertion anomaly**
4. **Deletion anomaly**

1. Repetition anomaly:

In repetition anomaly certain information may be repeated unnecessarily. Consider, for example, the EMP relation in slide 3 . The name, title, and salary of an employee are repeated for each project on which this person serves. This is obviously a waste of storage and is contrary to the spirit of databases.

2. Update anomaly:

In update anomaly, performing updates may be troublesome.

For example, if the salary of an employee changes, multiple tuples have to be updated to reflect this change.

3. Insertion anomaly:

It may not be possible to add new information to the database. For example, when a new employee joins the company, we cannot add personal information (name, title, salary) to the EMP relation unless an appointment to a project is made. This is because the key of EMP includes the attribute PNO, and null values cannot be part of the key.

4. Deletion anomaly:

This is the converse of the insertion anomaly. If an employee works on only one project, and that project is terminated, it is not possible to delete the project information from the EMP relation.

To do so would result in deleting the only tuple about the employee, thereby resulting in the loss of personal information we might want to retain.

Most Popular DBMS Query Languages

1. Oracle RDBMS
2. SolarWinds Database Performance Analyzer
3. IBM DB2
4. Altibase
5. Microsoft SQL Server
6. SAP Sybase ASE
7. Teradata
8. ADABAS
9. MySQL
10. FileMaker
11. Microsoft Access
12. Informix
13. SQLite
14. MariaDB
15. Informix Dynamic Server
16. 4D (4th Dimension)
17. PostgresSQL
18. AmazonRDS
19. MongoDB
20. Redis
21. CouchDB
22. Neo4j
23. OrientDB
24. Couchbase
25. Toad
26. phpMyAdmin
27. SQL Developer
28. Seqel PRO
29. Robomongo
30. DbVisualizer
31. Hadoop HDFS
32. Cloudera